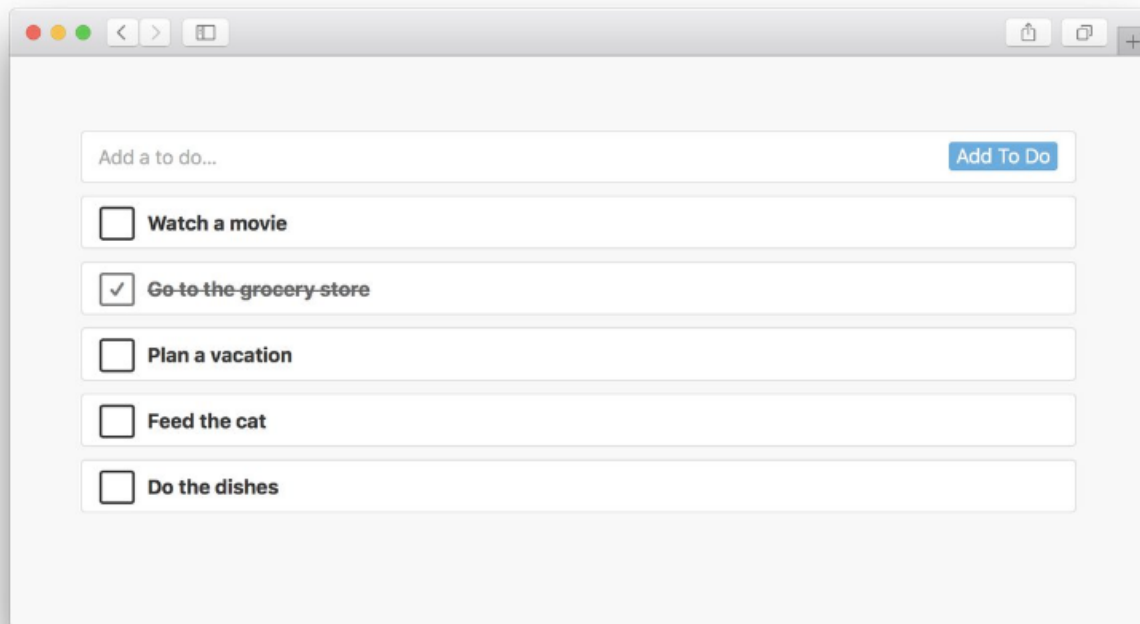


Take-Home Exercise: To-do List App



Objective

In this take-home project, you'll be building a simple to-do list app that fetches data from a server and saves data back to it.

- We recommend spending around 3 hours on this project.
- You're free to use Google, Stack Overflow, and other tools that you would normally use while programming.
- You're welcome to use any programming languages or open source libraries.
- Please visually match the mockups with pixel-perfect precision.
- Please do not consult or review your solution with others.

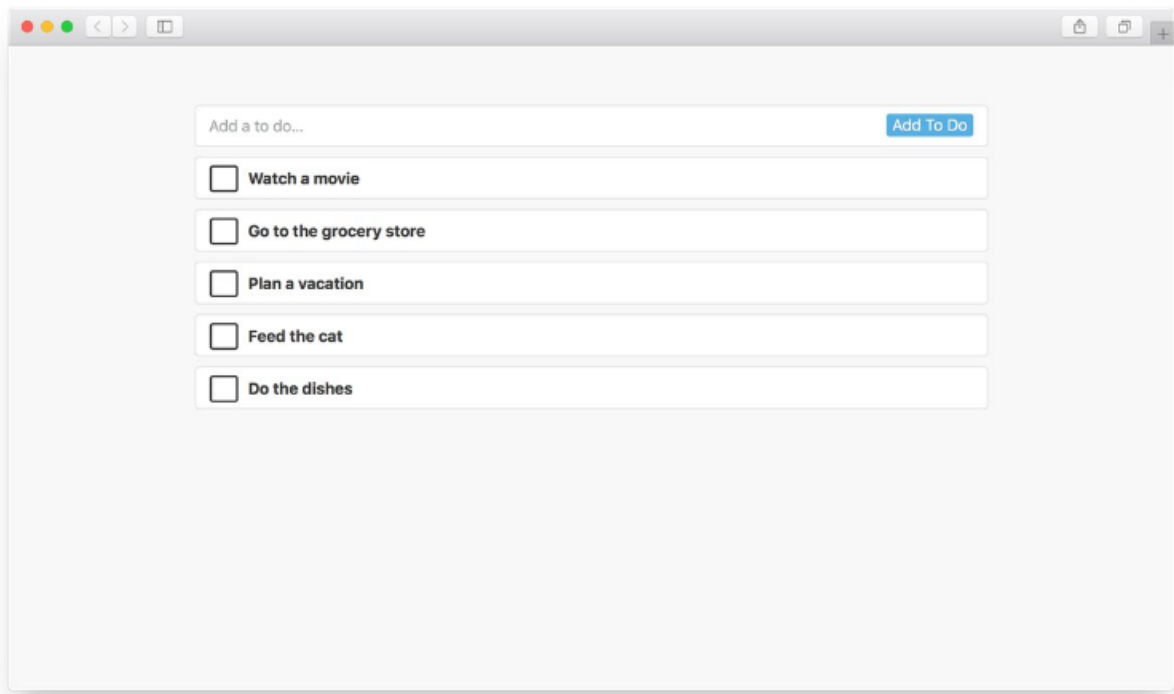
Features to Implement

1. Populate the to-do list with data fetched from the server, using the [Get All Todos](#) server endpoint.
2. Implement functionality to add a new to-do, which should also save it to the server using the [Add A New Todo](#) endpoint.
3. Implement functionality to mark to-dos as completed or not completed, using the [Mark Completion Of A Todo](#) endpoint.

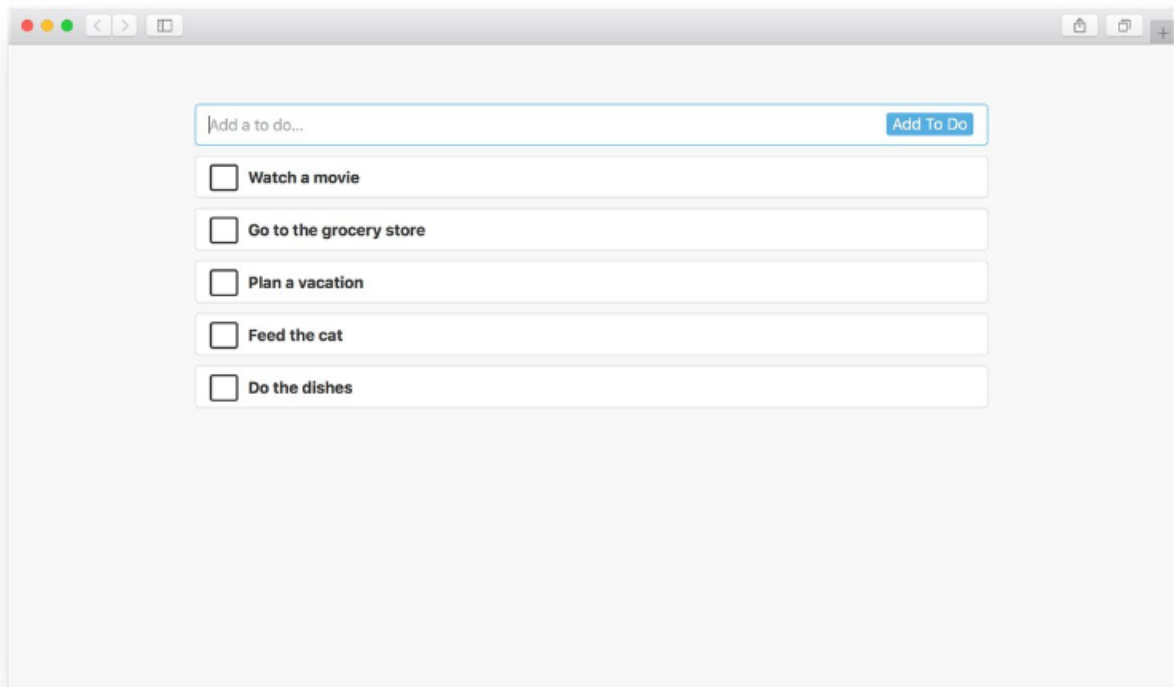
Submission Instructions

Once you're finished, please send us your code as well as instructions on how to run the app. You can send it as an email attachment, private GitHub link, or any other method you prefer. Feel free to also include details on any design decisions you made or other relevant notes.

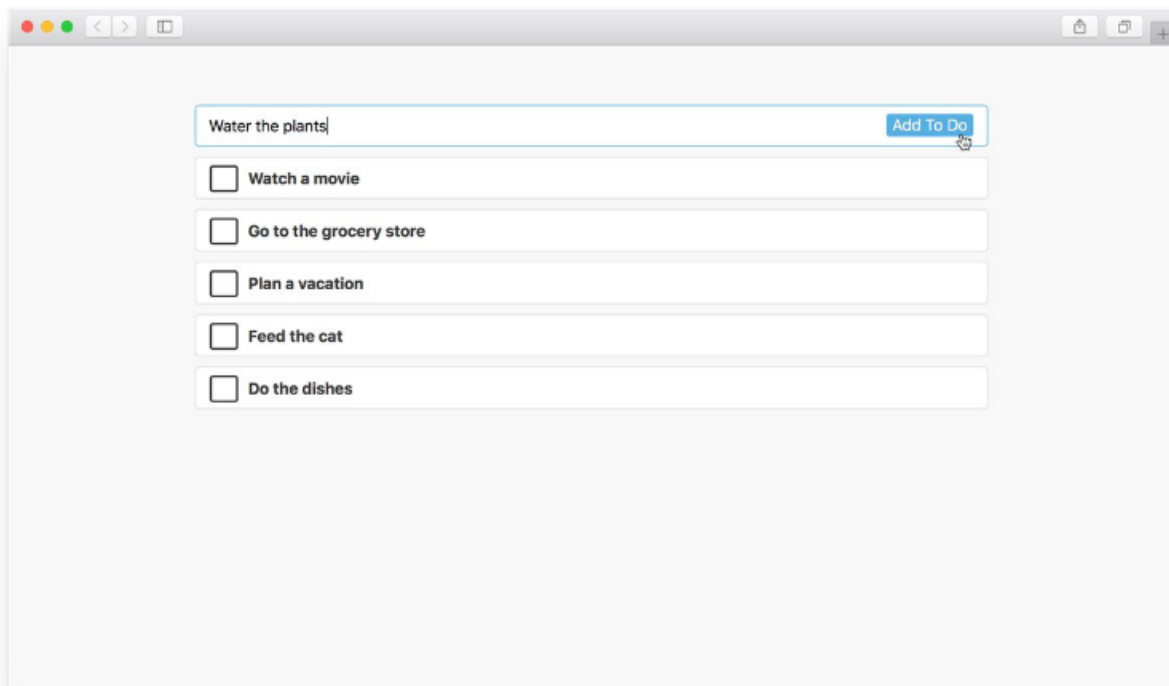
Mockups



A screenshot of a web application window. At the top, there is a header bar with standard macOS window controls (red, yellow, green buttons, and zoom/refresh icons) on the left and a share icon, a document icon, and a plus sign on the right. Below the header, the main content area has a light gray background. At the top of this area is a white input field with the placeholder text "Add a to do...". To the right of this field is a blue button with the text "Add To Do". Below the input field is a list of five items, each consisting of a small square checkbox followed by the item's text: "Watch a movie", "Go to the grocery store", "Plan a vacation", "Feed the cat", and "Do the dishes". Each item is contained within a thin white border.

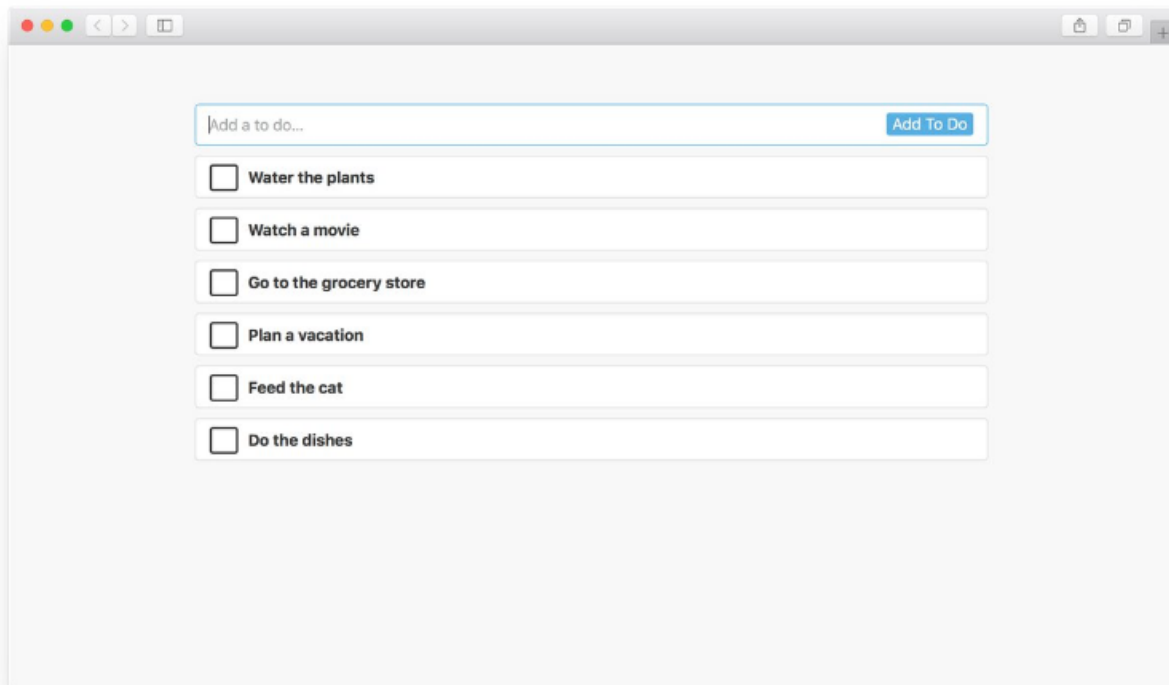


A second screenshot of the same web application window, showing the same interface as the first. The window controls and layout are identical. The input field at the top now contains a vertical cursor (text cursor) at the beginning, indicating it is active. The list of tasks remains the same: "Watch a movie", "Go to the grocery store", "Plan a vacation", "Feed the cat", and "Do the dishes".



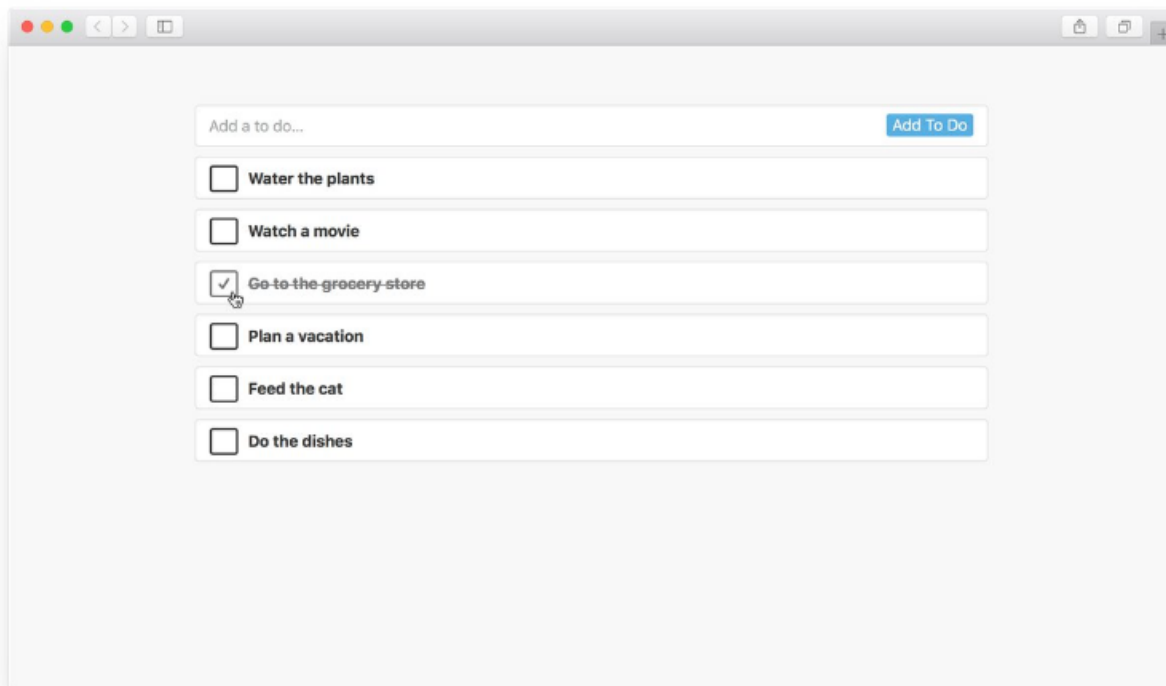
A screenshot of a web application window. At the top, there is a search bar containing the text "Water the plants" and a blue "Add To Do" button. Below the search bar, there is a list of five tasks, each with a checkbox and a text input field:

- ☐ Watch a movie
- ☐ Go to the grocery store
- ☐ Plan a vacation
- ☐ Feed the cat
- ☐ Do the dishes



A screenshot of the same web application window. The search bar now contains the placeholder text "Add a to do..." and the "Add To Do" button is still present. The list of tasks has been updated, with "Water the plants" added as the first item:

- ☐ Water the plants
- ☐ Watch a movie
- ☐ Go to the grocery store
- ☐ Plan a vacation
- ☐ Feed the cat
- ☐ Do the dishes



Server API Reference

Here is the documentation for the various endpoints on the server that your app will be communicating with. If you run into any issues with the server while working on the project, please email rajeev@quip.com.

GET ALL TODOS

This endpoint will return a JSON list of pre-populated todos, in addition to any todos you've added.

GET http://quip-todos.herokuapp.com/get_todos?email=example@gmail.com

Response:

```
[
  {
    id: 0,
    text: "Todo #1",
    completed: false
  },
  {
    id: 1,
    text: "Todo #2",
    completed: true
  },
  ...
]
```

ADD A NEW TODO

This endpoint will create a new todo with the provided text and return a JSON object representing the newly created todo.

POST http://quip-todos.herokuapp.com/add_todo

Body: email=example@gmail.com, text=Example Todo

Content-Type: application/x-www-form-urlencoded

Response:

```
{
  id: 2,
  text: "Example Todo",
  completed: false
}
```

MARK COMPLETION OF A TODO

This endpoint will change the 'completed' attribute of the todo with the given id and return a JSON object representing the modified todo.

POST http://quip-todos.herokuapp.com/mark_completed

Body: email=example@gmail.com, id=2, completed=true

Content-Type: application/x-www-form-urlencoded

Response:

```
{
  id: 2,
  text: "Example Todo",
  completed: true
}
```

7/2/2018

RESET BACK TO THE INITIAL TODOS

This endpoint will reset your data back to the initial set of todos, before you added any new ones. You'll only need to use this if you want to clear out the todos you added while testing.

POST http://quip-todos.herokuapp.com/reset

Body: email=example@gmail.com

Content-Type: application/x-www-form-urlencoded

Response:

"success"